152114002: NESNE TABANLI PROGRAMLAMA I
2016-2017 GÜZ DÖNEMİ
DÖNEM PROJESİ
Son Teslim Tarihi: 15 Ocak 2017, Pazar, 17:00

Açıklama ve Kurallar:

1. Grup çalışması içerisinde,
   i. Bütün grup üyeleri, tasarım sürecinde bulunmalıdır.
   ii. Görevler (sınıf kodlarının ve uygulamanın yazılması) tüm grup üyelerine dağıtılmalı ve raporda bu dağılım açıkça belirtilmelidir.
   iii. Her öğrenci, kendisine atana sınıf kodlarını ve test programlarını yazmalıdır. Her sınıf ayrı header (.h) ve ayrı source (.cpp) dosyasına sahip olmalıdır. Her dosyada kodu yazan kişinin ismi ve tarih bulunmalıdır.
2. Kodların yanında ilgili kod segmenti ile ilgili açıklamaların yazılması gerekmektedir. (Bakınız EK A.1).
3. Proje için, tasarım ve gerçekleme aşamasının aktarıldığı bir rapor hazırlanması gerekmektedir. (Bakınız EK A.2)
4. Nesne tabanlı yapıların kullanıldığı (abstraction, inheritance, polymorphism, templates, exception handling, etc) iyi bir tasarım ve kodlama yapmalısınız.
5. Proje için, yazılan kodları ve proje raporunu sıkıştırarak. zip ya da .rar olarak, ve dosyayı "Grup_Uyelerinden_birinin_ismi.zip/rar" şeklinde isimlendirip, DYS yüklemelisiniz.
6. Notlar, ekte verilen tabloya göre verilecektir. (Bakınız EK A.3)
7. Uygulama programı konsol tabanlı olmalıdır.
8. Sadece standart C++ kütüphaneleri kullanılmalıdır.
9. **<u>Tasarım ve/ve ya kodlarınızın kısmen ya da tamamen başka gruplardan ve referans gösterilmemiş kaynaklardan alındığı belirlenirse, sıfır not alırsınız.</u>**

PROJE BAŞLIĞI: Online Book Store

<u>Specifications:</u>

In this project, you will be implementing simple software for an online book store in C++. The book store will provide a shopping experience for your customers, while at the same time allowing you to practice developing object oriented coding as well as being a member of the team.

The class diagram which provides the framework for the book store is given in Figure 1.

Here's a brief description of each class:

- Product
    - An abstract item for sale in the store
    - Notable Attributes:
        - name - the name of the item
        - id - unique ID for the item
        - price - the price of the item
    - Possible Methods:
        - printProperties(): shows the all properties of the items.
- Book
    - An item for sale in a store
    - Notable Attributes:
        - author – the author of the book
        - publisher – the publisher of the book
        - page – the page number of the book
- Magazine
    - An item for sale in the store
    - Notable Attributes:
        - issue – the issue of the magazine
        - type – the type of the magazine (Actual, News, Sport, computer, technology, etc.)
- MusicCD
    - An item for sale in the store
    - Notable Attributes:
        - singer – the name of the singer
        - type – the type of the music (Romance, Hard Rock, Country, etc.)
- Customer
    - A customer!
    - Notable Attributes:
        - name - the name of the customer
        - customerID – unique id for each customer
        - bonus – each customer earns bonus as 1% of the amount of each shopping. Then, he/she may use his bonus next shopping or save it.
        - Username and password – a customer may shop after entering his username and password.
- ShoppingCart
    - A shopping cart for the customer
    - Notable Attributes:
        - itemsToPurchase – the list of itemsToPurchase currently in the shopping cart (use list or vector from STL)
        - payment – the payment method to purchase items
        - isBonusUsed – this has the value false as a default. If this set to true, customer pay shopping price discounted by his bonus.
    - Possible Methods:
        - printProducts() – shows the items currently in the cart

        - void addProduct(itemToPurchase) - add an item to the shopping cart
        - void removeProduct(itemToPurchase) - remove an item from the shopping cart
        - placeOrder() – perform the payment for the items in the cart and send invoice to custumer's email.
        - cancelOrder() – cancel the order
        - showInvoice() – show the invoice
- ItemToPurchase
    - A class includes an item and its quantity for adding to cart.
    - Notable Attributes:
        - product –an item to purchase
        - quantity – the quantity of the item to purchase
- Payment
    - An abstract for payment methods
    - Notable Attributes:
        - amount–an amount to pay
    - Possible Methods:
        - performPayment() – authorize the payment
- Credit
    - A payment method
    - Notable Attributes:
        - number– credit card number
        - type – type of the credit card (master/visa)
        - expDate – expiration date of the card
- Cash
    - A payment method
- Check
    - A payment method
    - Notable Attributes:
        - name– name of the check owner
        - BankID –bank id of the check

Note that required constructors, destructors are not listed in the above for the sake of brevity. You should add them.

Requirements:

Use the above framework to write a **working** version of the online book store.

Your application program must have at least five customers and three products from each type.

Implement the user interface any way you like, either textual or graphical.

Feel free to add new classes or/and new data and methods for given classes.

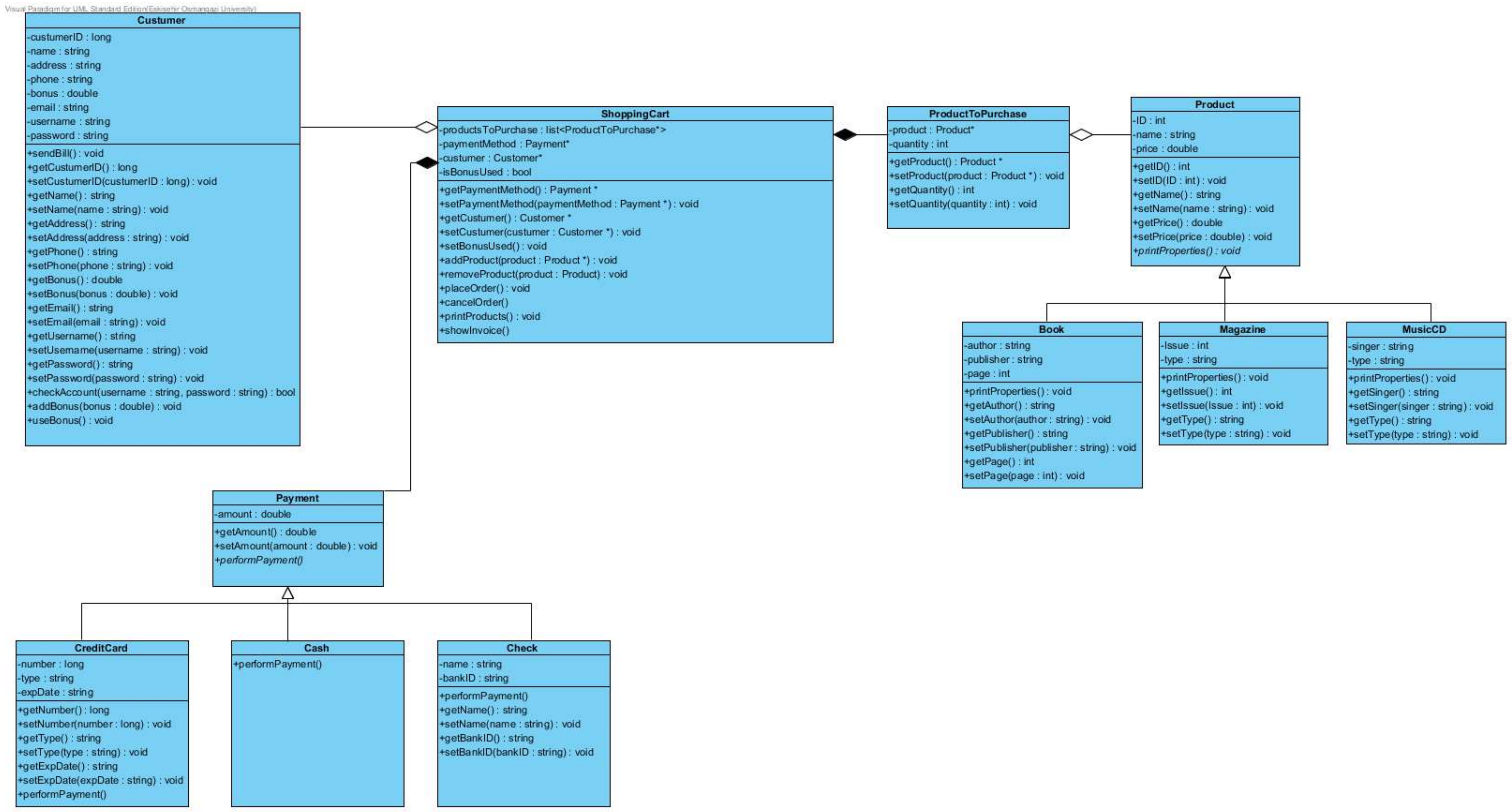

Visual Paradigm for UML Standard Edition(Eskisehir Osmangazi University)
Custumer
-custumerID : long
-name : string
-address : string
-phone : string
-bonus : double
-email : string
-username : string
-password : string
+sendBill() : void
+getCustumerID() : long
+setCustumerID(custumerID : long) : void
+getName() : string
+setName(name : string) : void
+getAddress() : string
+setAddress(address : string) : void
+getPhone() : string
+setPhone(phone : string) : void
+getBonus() : double
+setBonus(bonus : double) : void
+getEmail() : string
+setEmail(email : string) : void
+getUsername() : string
+setUsername(username : string) : void
+getPassword() : string
+setPassword(password : string) : void
+checkAccount(username : string, password : string) : bool
+addBonus(bonus : double) : void
+useBonus() : void
ShoppingCart
-productsToPurchase : list<ProductToPurchase*>
-paymentMethod : Payment*
-custumer : Customer*
-isBonusUsed : bool
+getPaymentMethod() : Payment *
+setPaymentMethod(paymentMethod : Payment *) : void
+getCustumer() : Customer *
+setCustumer(custumer : Customer *) : void
+setBonusUsed() : void
+addProduct(product : Product *) : void
+removeProduct(product : Product) : void
+placeOrder() : void
+cancelOrder()
+printProducts() : void
+showInvoice()
ProductToPurchase
-product : Product*
-quantity : int
+getProduct() : Product *
+setProduct(product : Product *) : void
+getQuantity() : int
+setQuantity(quantity : int) : void
Product
-ID : int
-name : string
-price : double
+getID() : int
+setID(ID : int) : void
+getName() : string
+setName(name : string) : void
+getPrice() : double
+setPrice(price : double) : void
+printProperties() : void
Book
-author : string
-publisher : string
-page : int
+printProperties() : void
+getAuthor() : string
+setAuthor(author : string) : void
+getPublisher() : string
+setPublisher(publisher : string) : void
+getPage() : int
+setPage(page : int) : void
Magazine
-Issue : int
-type : string
+printProperties() : void
+getIssue() : int
+setIssue(Issue : int) : void
+getType() : string
+setType(type : string) : void
MusicCD
-singer : string
-type : string
+printProperties() : void
+getSinger() : string
+setSinger(singer : string) : void
+getType() : string
+setType(type : string) : void
Payment
-amount : double
+getAmount() : double
+setAmount(amount : double) : void
+performPayment()
CreditCard
-number : long
-type : string
-expDate : string
+getNumber() : long
+setNumber(number : long) : void
+getType() : string
+setType(type : string) : void
+getExpDate() : string
+setExpDate(expDate : string) : void
+performPayment()
Cash
+performPayment()
Check
-name : string
-bankID : string
+performPayment()
+getName() : string
+setName(name : string) : void
+getBankID() : string
+setBankID(bankID : string) : void

Figure 1. UML Class Diagram of the Project

A Sample User Interface
You can also design the user interface menus as objects. I really like it, and give reward.

```
Main Menu
1. Customer
2. Item
3. Shopping
4. Quit
Choose one : 1 ↵

Customer Menu
1. Add a new customer to system
2. Show the customers in the system
3. Back
Choose one : 1 ↵

Add Customer
Customer Name: Ali Kaya ↵
Customer Address: Batıkent Mah. No: 4, Eskisehir ↵
          .
          .
          .

Customer Menu
1. Add a custumer item to system
2. Show the custumer in the system
3. Back
Choose one : 3 ↵

Main Menu
1. Customer
2. Item
3. Shopping
4. Quit
Choose one : 3 ↵

Shopping Menu
1. Login
2. Add Product
3. Remove Product
4. List All Products
5. List Shopping Cart
6. Show Bonus
7. Use Bonus
8. place Order
9. cancel order
10. Show invoice
11. Quit
Choose one : 1 ↵

Enter the usename: akaya
Enter the password: 12345
          .
          .
```

Appendix:

A.1 Doxygen HowTo

Doxygen is a documentation system for C++, C, Java, Objective-C, Python, IDL (Corba and Microsoft flavors), Fortran, VHDL, PHP, C#, and to some extent D.

It can help you in three ways:

1. It can generate an on-line documentation browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files. There is also support for generating output in RTF (MS-Word), PostScript, hyperlinked PDF, compressed HTML, and Unix man pages. The documentation is extracted directly from the sources, which makes it much easier to keep the documentation consistent with the source code.
2. You can configure doxygen to extract the code structure from undocumented source files. This is very useful to quickly find your way in large source distributions. You can also visualize the relations between the various elements by means of include dependency graphs, inheritance diagrams, and collaboration diagrams, which are all generated automatically.
3. You can also use doxygen for creating normal documentation (as I did for this manual).

Doxygen is developed under Linux and Mac OS X, but is set-up to be highly portable. As a result, it runs on most other Unix flavors as well. Furthermore, executables for Windows are available.

For more detail, http://www.stack.nl/~dimitri/doxygen/manual.html

Sample :

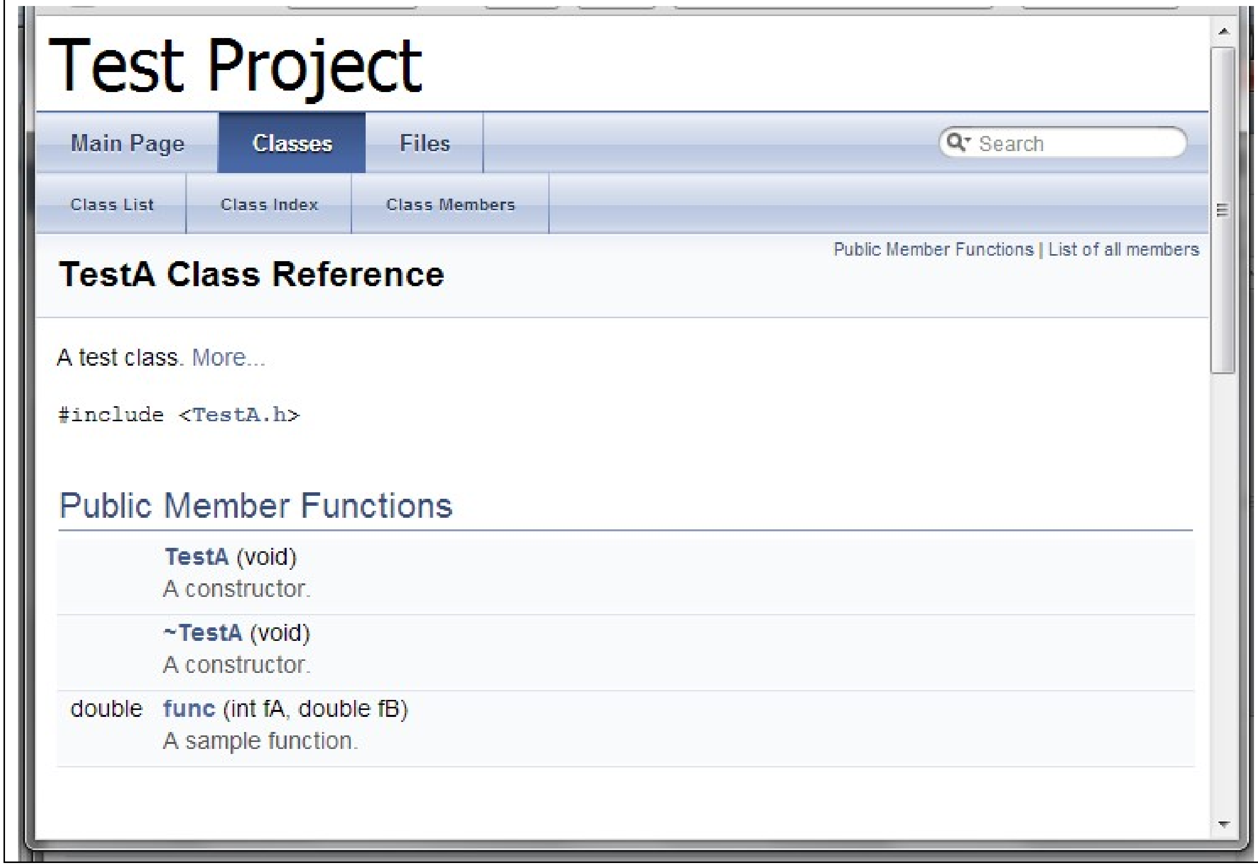
The Header File (TestA.h) is:

```
/**
 * @file   TestA.h
 * @Author Me (me@example.com)
 * @date   September, 2008
 * @brief  Brief description of file.
 *
 * Detailed description of file.
 */

//! An enum.
/*! More detailed enum description. */
enum TestENUM{
     ENUM1,  /*!< Definition of ENUM1 */
     ENUM2,    /*!< Definition of ENUM2 */
     ENUM3     /*!< Definition of ENUM3 */
};

//!  A test class.
/*!
  A more elaborate class description.
*/
class TestA{
public:
     //! A constructor.
     TestA(void);
     //! A constructor.
     ~TestA(void);
     //! A sample function.
     double func(int fA, double fB);
};
```

The Source File (TestA.cpp) is:

```
#include "TestA.h"

TestA::TestA(void)
{}

TestA::~TestA(void)
{}

/*!
     \param fA an integer argument.
    \param fB an double argument.
    \return The test results
*/
double TestA::func(int fA, double fB)
{
     return fA*fB;

}
```

Snapshots from html type documentation pages after generating by Doxygen



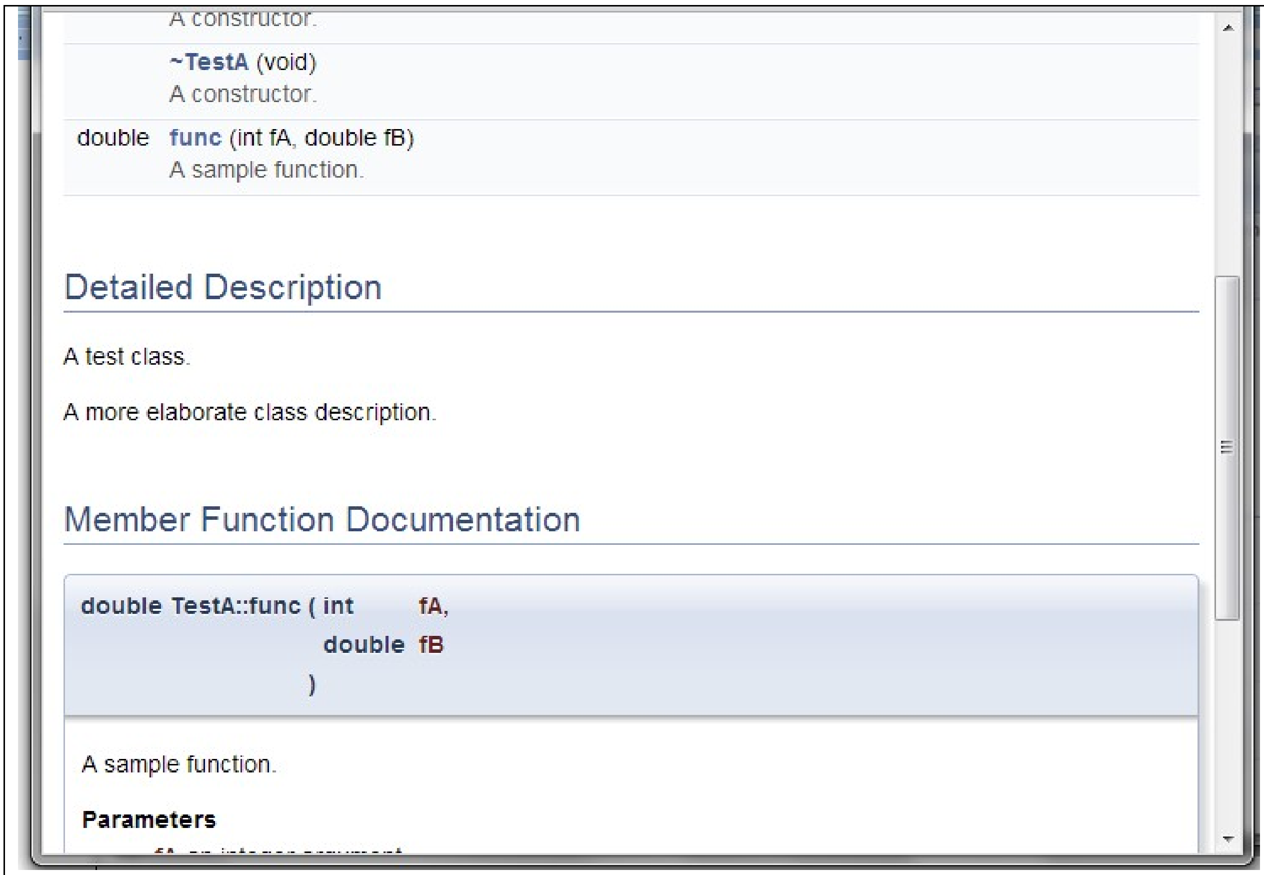
TestA Class Reference

A test class. More...

`#include <TestA.h>`

Public Member Functions

	TestA (void) A constructor.
	~TestA (void) A constructor.
double	**func** (int fA, double fB) A sample function.



A constructor.

~TestA (void)
A constructor.

double func (int fA, double fB)
A sample function.

Detailed Description

A test class.

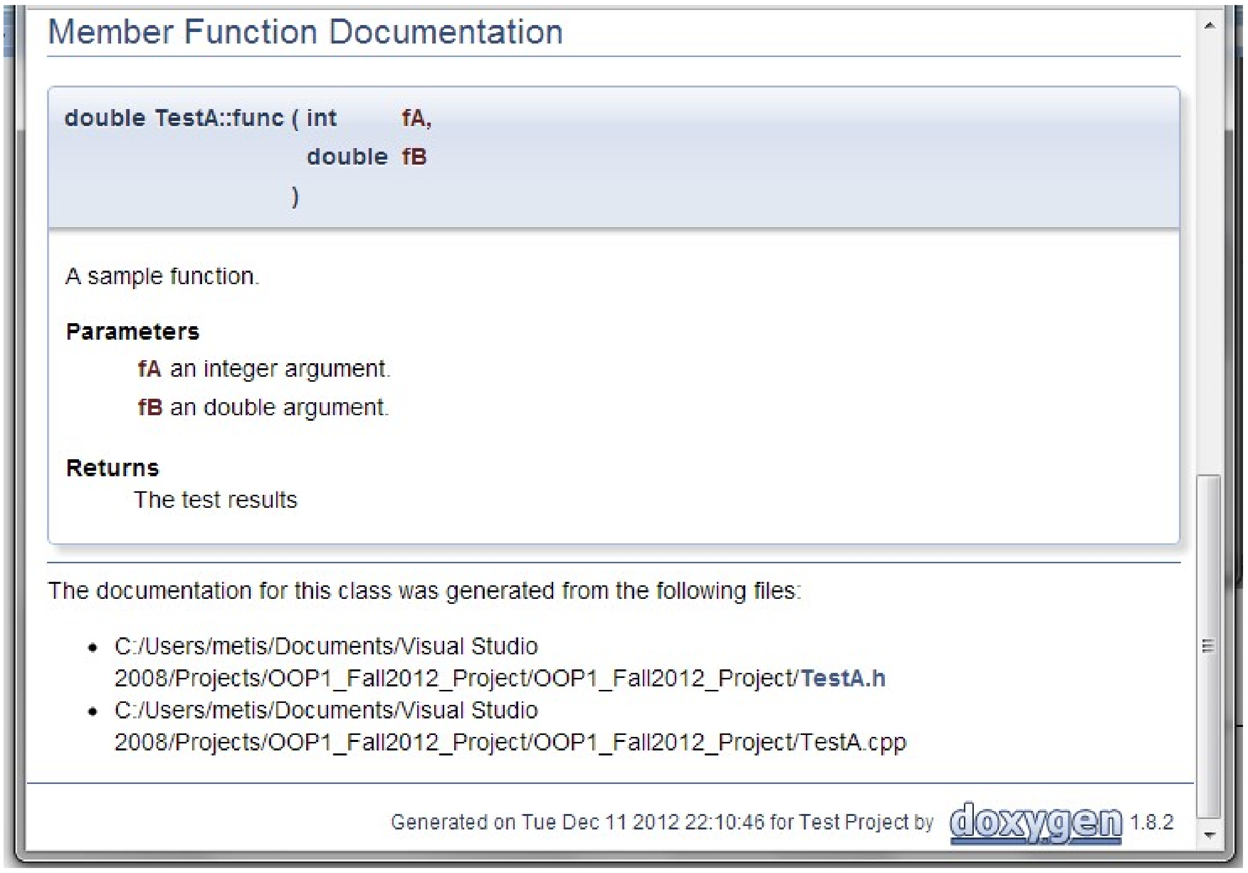
A more elaborate class description.

Member Function Documentation

```
double TestA::func ( int     fA,
                     double  fB
                   )
```

A sample function.

Parameters



Member Function Documentation

```
double TestA::func ( int     fA,
                     double  fB
                   )
```

A sample function.

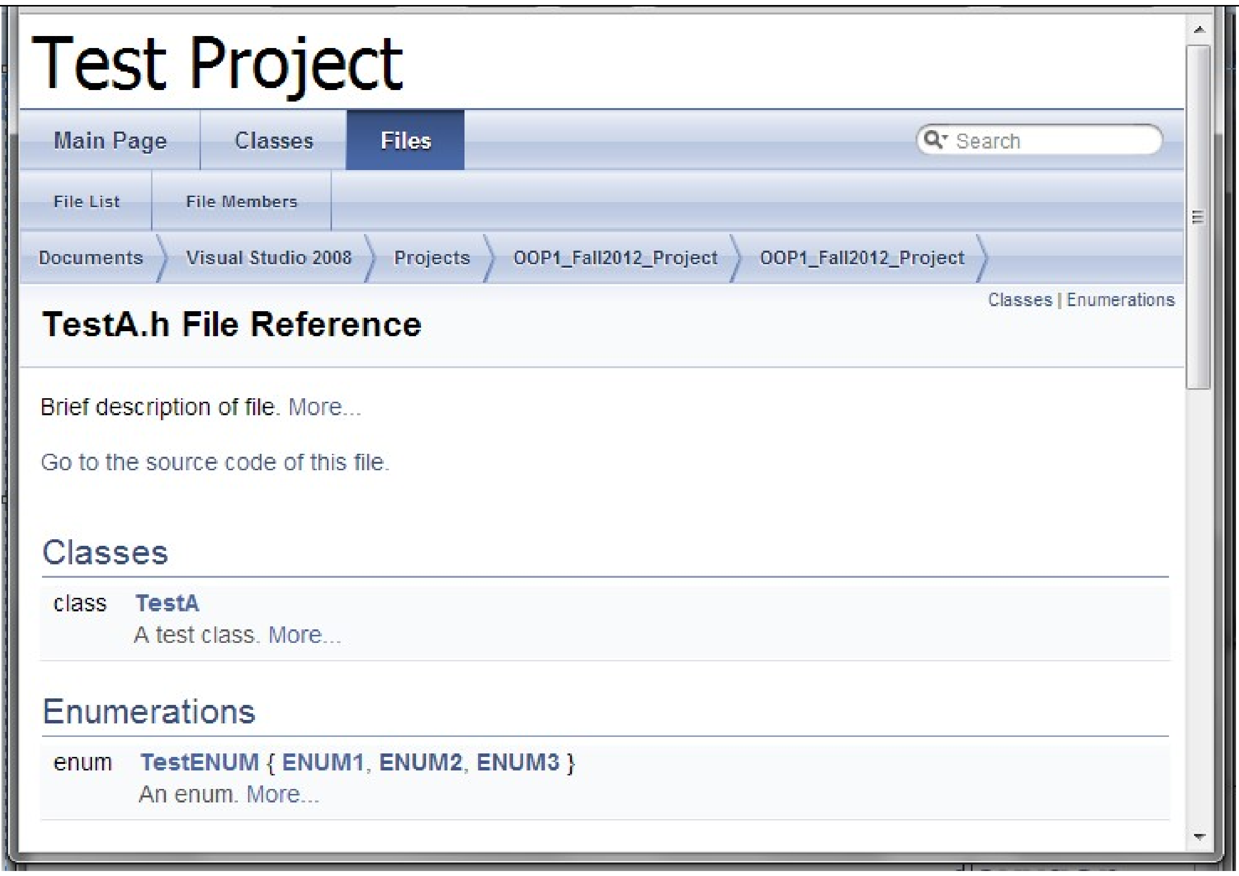
Parameters
- **fA** an integer argument.
- **fB** an double argument.

Returns
The test results

The documentation for this class was generated from the following files:

- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/**TestA.h**
- C:/Users/metis/Documents/Visual Studio 2008/Projects/OOP1_Fall2012_Project/OOP1_Fall2012_Project/TestA.cpp

Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2



Test Project

Main Page | Classes | Files | Search

File List | File Members

Documents > Visual Studio 2008 > Projects > OOP1_Fall2012_Project > OOP1_Fall2012_Project

Classes | Enumerations

TestA.h File Reference

Brief description of file. More...

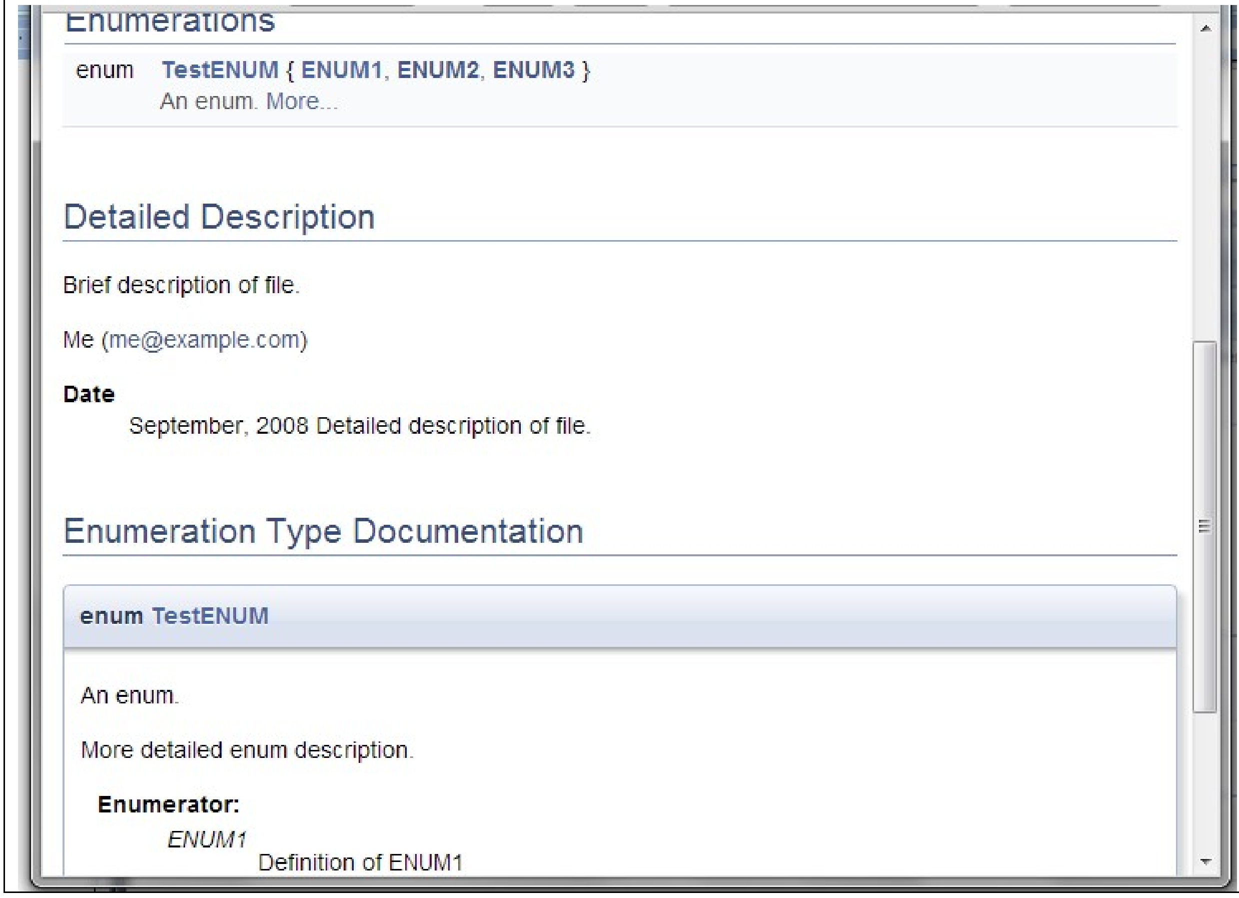
Go to the source code of this file.

Classes

class **TestA**
A test class. More...

Enumerations

enum **TestENUM** { **ENUM1**, **ENUM2**, **ENUM3** }
An enum. More...



Enumerations

enum **TestENUM** { **ENUM1**, **ENUM2**, **ENUM3** }
An enum. More...

Detailed Description

Brief description of file.

Me (me@example.com)

Date
September, 2008 Detailed description of file.

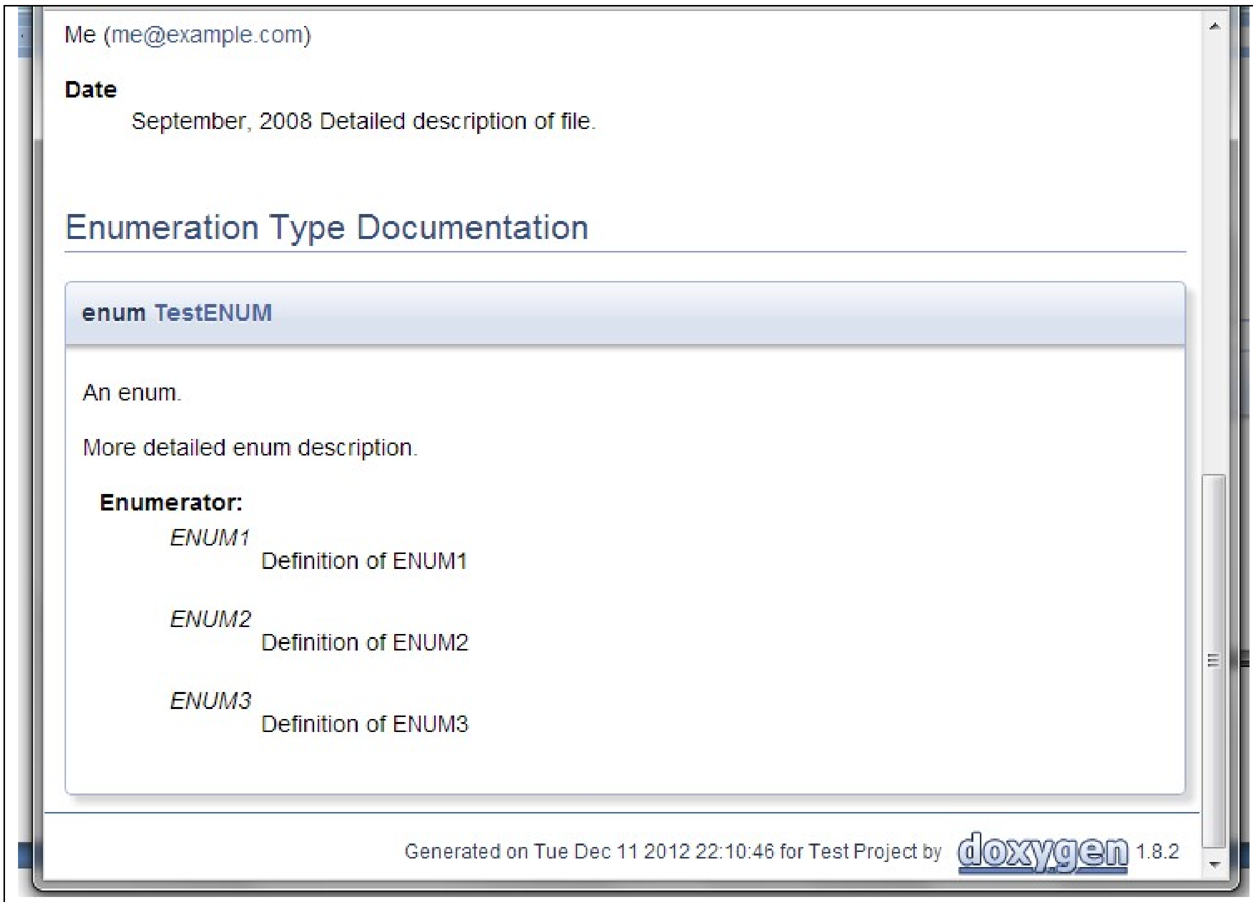
Enumeration Type Documentation

enum TestENUM

An enum.

More detailed enum description.

Enumerator:
ENUM1 Definition of ENUM1



Me (me@example.com)

Date
September, 2008 Detailed description of file.

Enumeration Type Documentation

enum TestENUM

An enum.

More detailed enum description.

Enumerator:

ENUM1
Definition of ENUM1

ENUM2
Definition of ENUM2

ENUM3
Definition of ENUM3

Generated on Tue Dec 11 2012 22:10:46 for Test Project by doxygen 1.8.2

A.2 Report Format

Cover Page

ESKISEHIR OSMANGAZI UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

OBJECT ORIENTED PROGRAMMING I
DESIGN PROJECT REPORT

Project Title

Project Group Member ID and Name
Project Group Member ID and Name
Project Group Member ID and Name
...

January 2015

Report Contents

1. **Introduction**
   General information about project

2. **Design**
   You may add subtitles. This part includes explanations about design, UML diagrams, task assignments, sample outputs of the application program for the sample inputs, etc.

Task Assignments (Must be included)

Group Member	**Tasks**
Name of Group member	*Class(es),menus, documentation, etc. . . .*
Name of Group member	. . .
. . .	. . .

3. **Conclusion**
   Evaluation of the project results, comments about the team work, pros and cons, advices for further works, etc.

A.3 Check list for the materials which should be submitted and grading

CHECK LIST

Materials	**Included**
Header and source files of each classes and application (.h, .cpp)	Yes / No
Each class has separate header and source files?	Yes / No
Codes are well formatted?	Yes / No
Codes are commented by using Doxygen tags?	Yes / No
There are test programs for each class?	Yes / No
Internal documentation generated by Doxygen (.htm)	Yes / No
Report	Yes / No

GRADING (TENTATIVE)

Items	**Grade (%)**
Individual Studies	
Internal Documentation	10
Code quality (well formatted)	10
Completeness	10
Overall (including all classes and application)	
Internal documentation (using doxygen)	15
Overall code quality	10
Group work	15
Functionality (implemented and correctly running functionalities).	10
Report	20